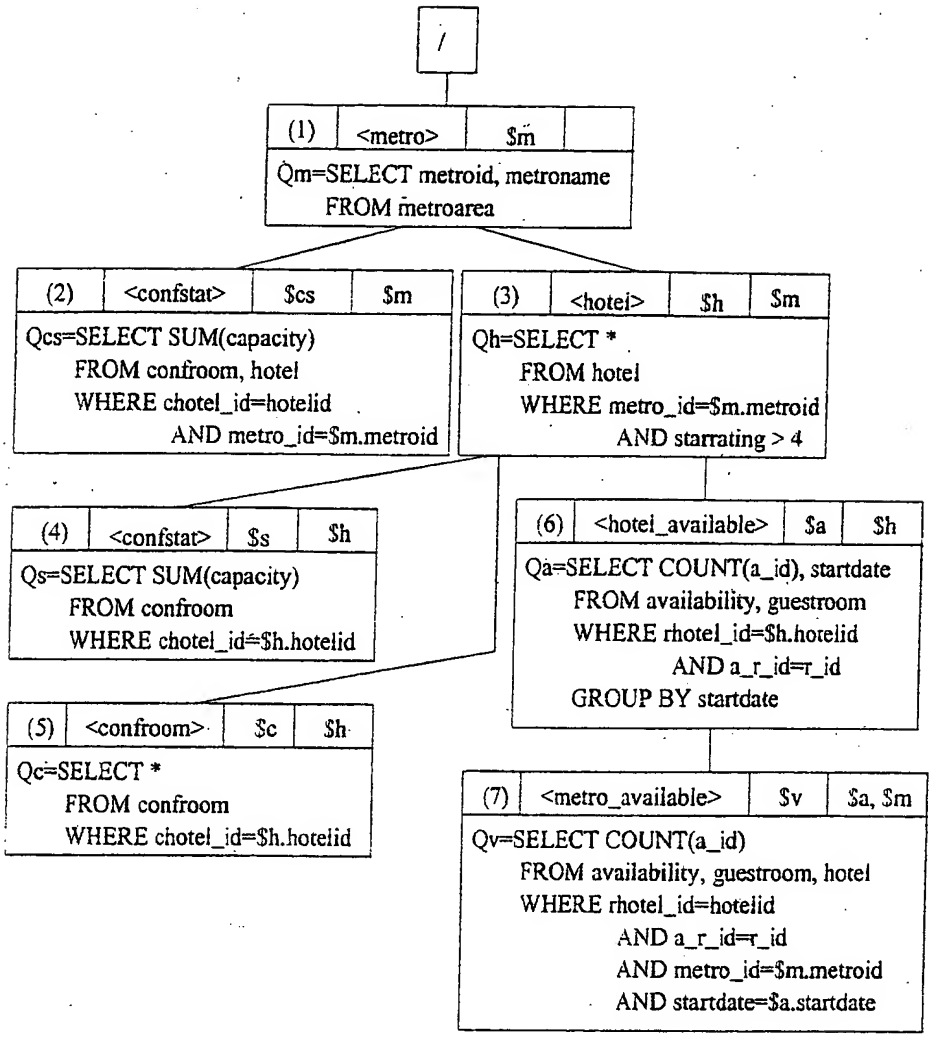

/
(1) <metro> $m
Qm=SELECT metroid, metroname
FROM metroarea
(2) <confstat> $cs $m
Qcs=SELECT SUM(capacity)
FROM confroom, hotel
WHERE chotel_id=hotelid
AND metro_id=$m.metroid
(3) <hotel> $h $m
Qh=SELECT *
FROM hotel
WHERE metro_id=$m.metroid
AND starrating > 4
(4) <confstat> $s $h
Qs=SELECT SUM(capacity)
FROM confroom
WHERE chotel_id=$h.hotelid
(5) <confroom> $c $h
Qc=SELECT *
FROM confroom
WHERE chotel_id=$h.hotelid
(6) <hotel_available> $a $h
Qa=SELECT COUNT(a_id), startdate
FROM availability, guestroom
WHERE rhotel_id=$h.hotelid
AND a_r_id=r_id
GROUP BY startdate
(7) <metro_available> $v $a, $m
Qv=SELECT COUNT(a_id)
FROM availability, guestroom, hotel
WHERE rhotel_id=hotelid
AND a_r_id=r_id
AND metro_id=$m.metroid
AND startdate=$a.startdate

FIG. 1

hotelchain(chainid, companyname, hqstate)

metroarea(metroid, metroname)

hotel(hotelid, hotelname, starrating, chain id

metro id, state id, city, pool, gym)

guestroom(r id, rhotel id, roomnumber, type, rackrate)

confroom(c id, chotel id, croomnumber, capacity, rackrate)

availability(a id, a r id, startdate, enddate, price)

FIG. 2


```
<metro name="NYCMetroNJ">
    <hotel name="Hyton" pool="Y" gym="N">
        <confstat capacity="3200"/>
        <confroom name="Green Flamingo" capacity="240"/> ...
        ...
    </hotel>
    <hotel> ... </hotel> ...
</metro>
```

FIG. 3

400

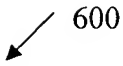
```
<xsl:template match="pattern" mode=”integer”>
   <Result>
        <xsl:apply-templates select="expression"/>
   </Result>
</xsl:template>
```

FIG. 4

```
<xsl:template match="/" >                                    (R1)
  <HTML>
      <HEAD></HEAD>
      <BODY>
          <xsl:apply-templates select="metro"/>
      </BODY>
  </HTML>
</xsl:template>
<xsl:template match="metro">                                 (R2)
  <result_metro>
    <A></A>
    <xsl:apply-templates
      select="hotel/confstat"/>
  </result_metro>
</xsl:template>
<xsl:template match="confstat">                              (R3)
  <result_confstat>
    <B></B>
    <xsl:apply-templates
     select="../hotel_available/../confroom"/>
  </result_confstat>
</xsl:template>

<xsl:template match="metro/hotel/confroom">                  (R4)
  <xsl:value-of select="."/>
</xsl:template>
```

FIG. 5



Function *PROCESS*(x, d_{con}, *mode*)

Input: XSLT stylesheet **as** x, Context document node **as** d_{con} ,
Desired mode of the matched rule **as** *mode*

Output: XML document fragment **as** *result*

1: *result* ← *empty*
2: **for** $r_i \in x$ in decreasing order of *priority*(r_i) **do**
3: **if** *mode*(r_i) = *mode* and *MATCH*(d_{con}, r_i) **then**
4: *result* ← *output* (r_i)
5: **for** $a_j \in$ *apply*(r_i) **do**
6: D_{new_con} ← *SELECT*(d_{con}, a_j)
7: *sub_result* ← *empty*
8: **for** $d_{new\_con} \in D_{new\_con}$ **do**
9: *sub_result* ← *concatenate*(*sub_result*, *PROCESS*(x, d_{new_con}, *mode*(a_j)))
10: Replace a_j in *result* with *sub_result*
11: **return** *result*

FIG. 6

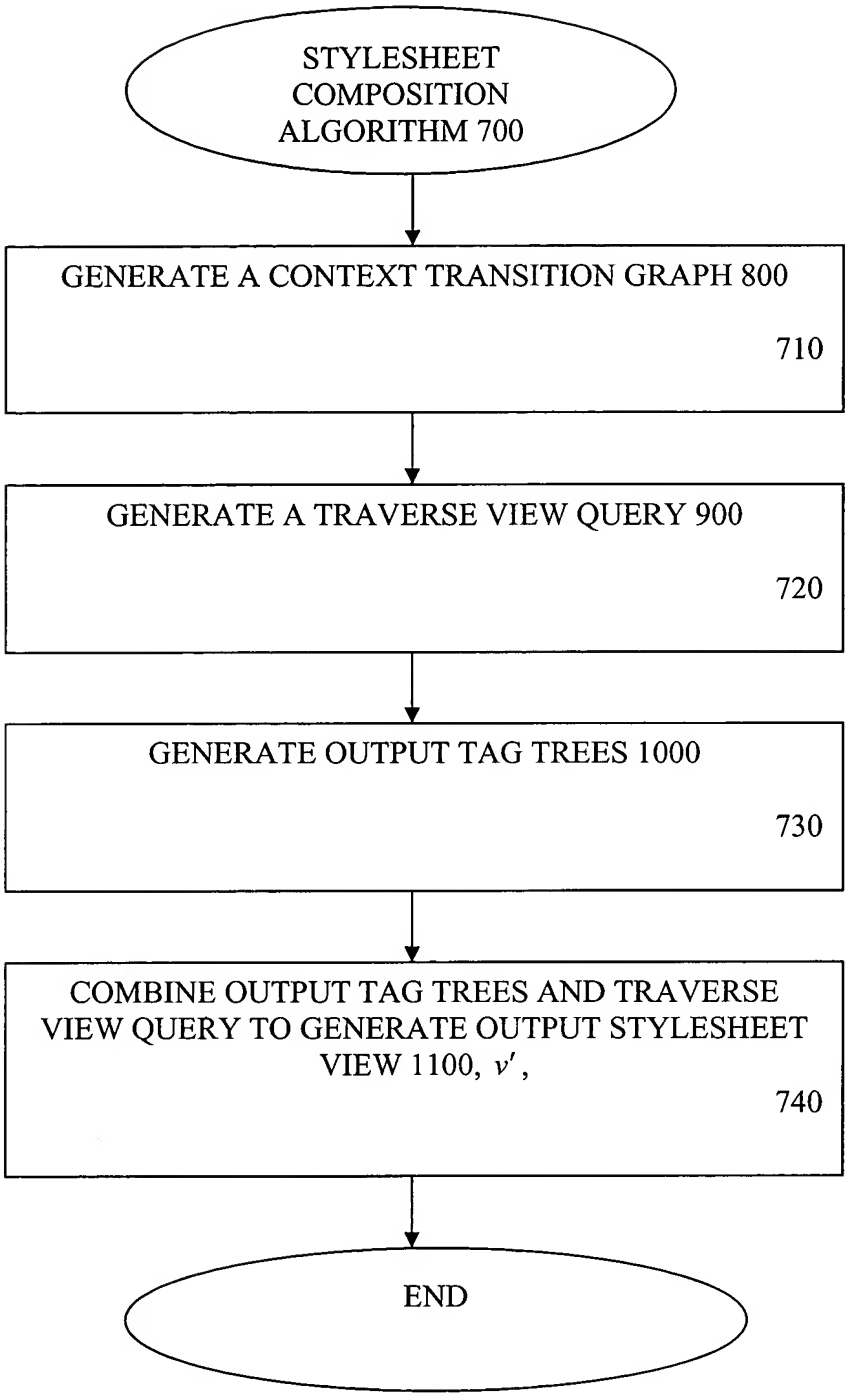

STYLESHEET COMPOSITION ALGORITHM 700
GENERATE A CONTEXT TRANSITION GRAPH 800
710
GENERATE A TRAVERSE VIEW QUERY 900
720
GENERATE OUTPUT TAG TREES 1000
730
COMBINE OUTPUT TAG TREES AND TRAVERSE VIEW QUERY TO GENERATE OUTPUT STYLESHEET VIEW 1100, v',
740
END

FIG. 7

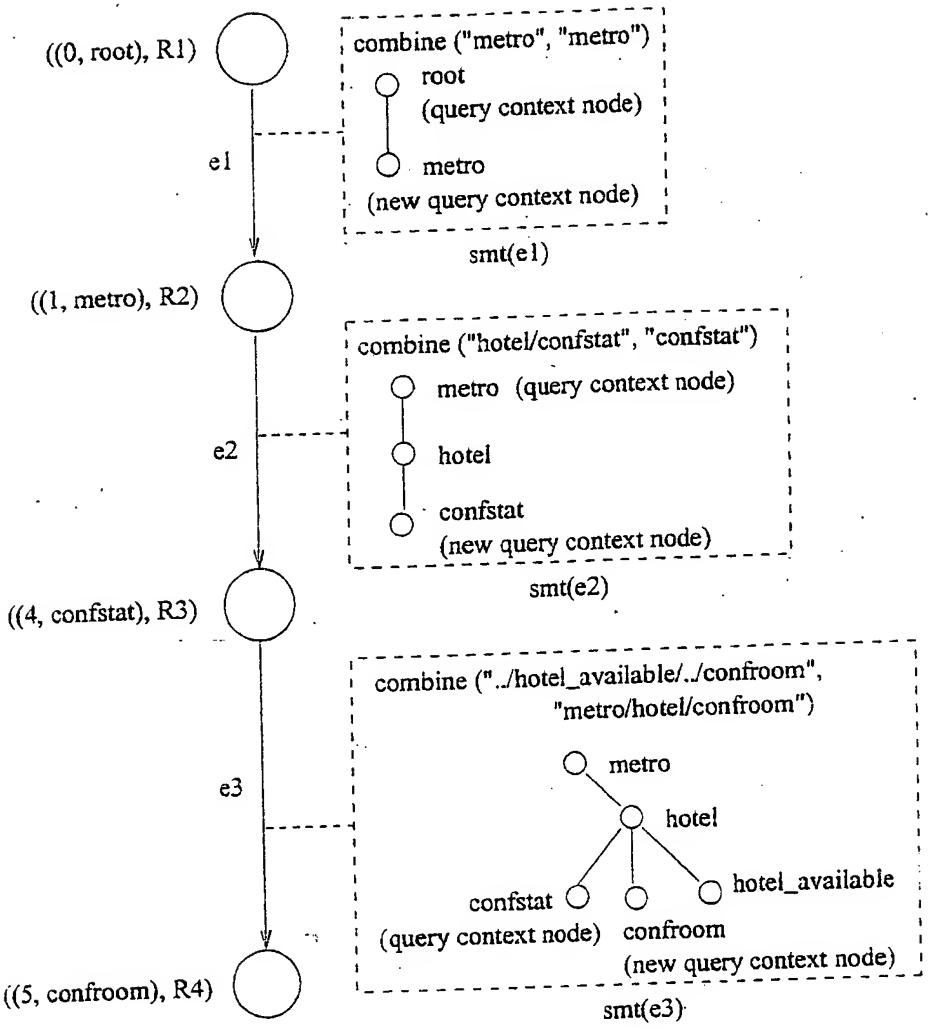

((0, root), R1)
e1
combine ("metro", "metro")
root
(query context node)
metro
(new query context node)
smt(e1)
((1, metro), R2)
e2
combine ("hotel/confstat", "confstat")
metro (query context node)
hotel
confstat
(new query context node)
smt(e2)
((4, confstat), R3)
e3
combine ("../hotel_available/../confroom",
"metro/hotel/confroom")
metro
hotel
confstat
hotel_available
(query context node)
confroom
(new query context node)
smt(e3)
((5, confroom), R4)

FIG. 8

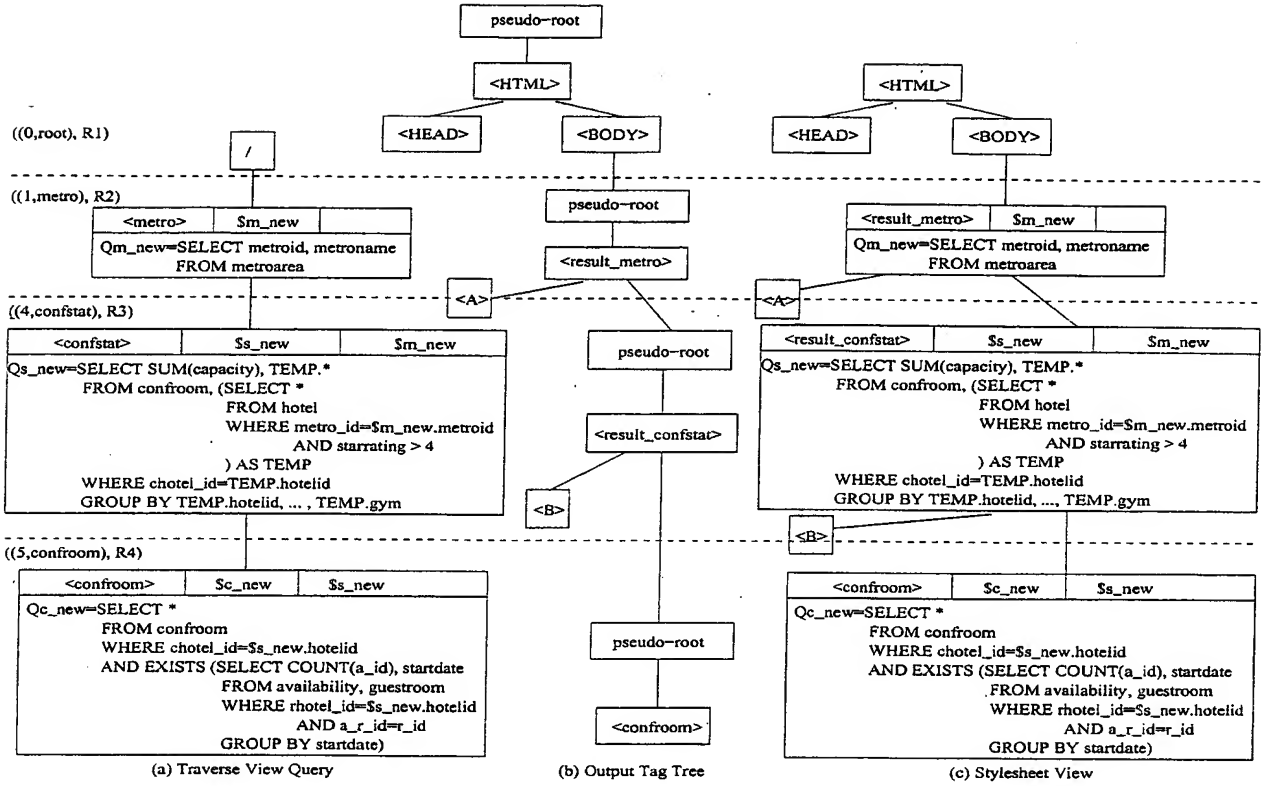


FIG. 9 FIG. 10 FIG. 11

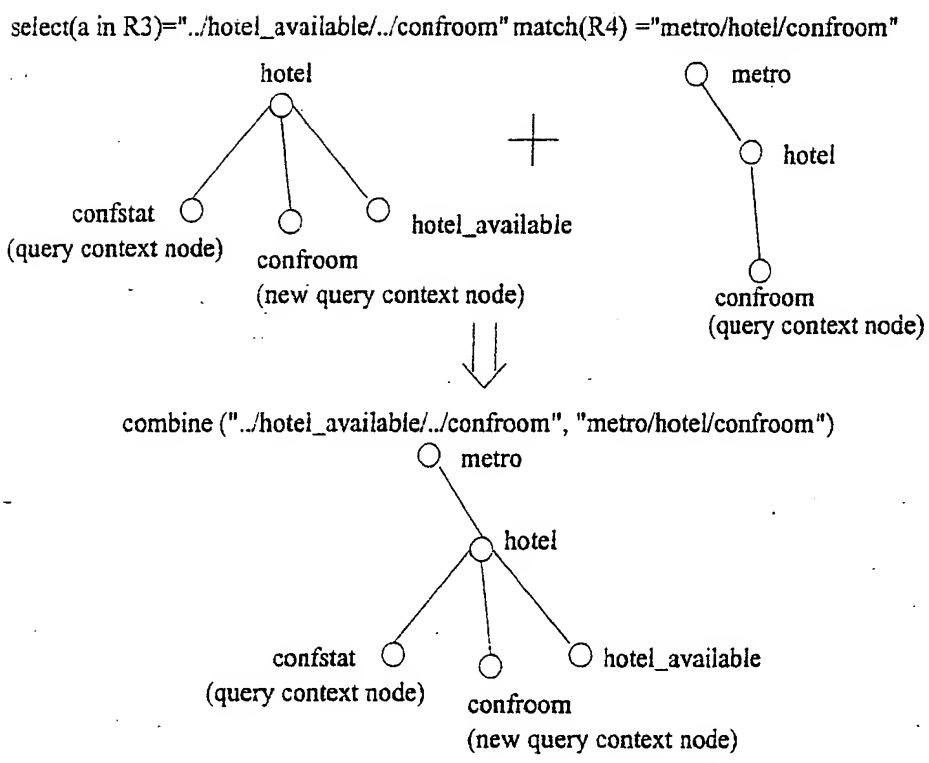

select(a in R3)="../hotel_available/../confroom" match(R4) ="metro/hotel/confroom"
hotel
confstat
(query context node)
confroom
(new query context node)
hotel_available
metro
hotel
confroom
(query context node)
combine ("../hotel_available/../confroom", "metro/hotel/confroom")
metro
hotel
confstat
(query context node)
confroom
(new query context node)
hotel_available

FIG. 12

Procedure *Compose*(v,x)
Input: v: original schema-tree view query; x: XSLT stylesheet
Output: stylesheet view

ctg: a Context Transition Graph
tvq: a Traverse View Query
ottree: an Output Template Tree
for $n \in v$ **do**
  for $r \in x$ **do**
    if $MATCHQ(n, r) \neq$ NULL **then**
      add (n, r) to *ctg*
for $(n_1, r_1) \in ctg$ **do**
  for $(n_2, r_2) \in ctg$ **do**
    for $a \in apply(r_1)$ **do**
      $t \leftarrow SELECTQ(n_1, a, n_2)$, $p \leftarrow MATCHQ(n_2, r_2)$
      if $t \neq$ NULL **and** $mode(a) = mode(r_2)$ **then**
        add an edge $e = ((n_1, r_1), (n_2, r_2), a)$ to *ctg*
        $smt(e) \leftarrow COMBINE(t, p)$
(repeatedly) Delete all nodes without incoming edge, except $(root, r)$
tvq $\leftarrow$ *ctg* {(copy)}, *bvmap*(root of *tvq*) $\leftarrow$ empty
(repeatedly) Duplicate nodes with multiple incoming edges, splitting incoming edges and copying outgoing edges
replace binding variables in *tvq* with new, unique binding variables
for $e = (w_1 = (n_1, r_1), w_2 = (n_2, r_2), a)$ in edges of *ctg* **do**
  $(\mathcal{Q}_{bv(w_2)}, bvmap(w_2)) \leftarrow$
  $UNBIND(smt(e), n_1, n_2, bv(w_2), bvmap(w_1))$
  for all binding variables *bv* referenced in $\mathcal{Q}_{bv(w_2)}$ **do**
    rename *bv* as $bvmap(w_2).get(bv)$
for $w = (n, r) \in tvq$ **do**
  $ott(w) = GENERATE\_OTT(n, r)$
ottree $\leftarrow ott(w) \forall w \in tvq$ {initially a forest}
for $e = (w_1, w_2, a)$ in edges of *tvq* **do**
  $a' \leftarrow$ the "apply-template" node in $ott(w_1)$ which is a copy of a
  replace a' with an edge from $parent(a')$ to $root(ott(w_2))$
for $w = (n, r) \in tvq$ **do**
  $bv(root(ott(w))) \leftarrow bv(w)$
  $\mathcal{Q}_{bv(root(ott(w)))} \leftarrow \mathcal{Q}_{bv(w)}$
Remove the topmost "pseudo-root" node in *ottree*
while there is any "pseudo-root" node *pr* left **do**
  for each child node *c* of *pr* **do**
    add edge $e = (parent(pr), c)$
    if $\mathcal{Q}_{bv(c)}$ is empty **then**
      $bv(c) \leftarrow bv(pr)$, $\mathcal{Q}_{bv(c)} \leftarrow \mathcal{Q}_{bv(pr)}$
    else
      $\mathcal{Q}_{bv(c)} \leftarrow UNBIND(parent(pr), c)$
      add the *SELECT* columns of $\mathcal{Q}_{bv(pr)}$ to $\mathcal{Q}_{bv(c)}$
      change "$bv(pr)$" as "$bv(c)$" in the tag queries of *c*'s descendents
  remove edge $e = (parent(pr), pr)$ and node *pr*
return *ottree* {new stylesheet view}

FIG. 13

Function $UNBIND(m, n)$
Input: Query context node as m, Target node to unbind as n
Output: Unbound query for n as q
1: $n_j \leftarrow$ the lowest common ancestor of m and n
2: $s_j \leftarrow bv(n_j)$
3: $child_n(n_j) \leftarrow$ child node of n_j along the path from n_j to n
4: $s_{j-1} \leftarrow bv(child_n(n_j))$
5: $q \leftarrow Q_{bv(n)}^{s_1,s_2,...,s_{j-1}}(s_j, ..., s'_k)$, which is the unbound tag query of n, $Q_{bv(n)}(s_1, s_2, ..., s_k)$
6: **return** q

FIG. 14

Function $NEST(p, p')$
Input: Node as p, Child node as p'
Output: Tag query for p after nesting as q
1: $q \leftarrow Q_{bv(p)}$
2: **for** each child node c of p, except p' **do**
3: $q_c \leftarrow NEST(c, NULL)$
4: add $EXISTS$ q_c in $WHERE$ clause of q
5: **return** q

FIG. 15

1: {**Replace line 5 of Figure 10**}
2: $P \leftarrow$ {nodes along the path from $child_n(n_j)$ to n}
3: **for** node $p \in P$ **do**
4: $p' \leftarrow$ child of $p \in P$, otherwise $NULL$
5: $\Theta_{bv(p)} \leftarrow NEST(p, p')$
6: $q \leftarrow \Theta_{bv(n)}^{s_1,s_2,...,s_{j-1}}(s_j, ..., s'_k)$, which is the unbound and nested tag query of n, $\Theta_{bv(n)}(s_1, s_2, ..., s_k)$, and decorrelation of $bv(s_i)$ is done by Θ_{s_i}.

FIG. 16

Function $UNBIND(smt, m, n, bv', bvmap)$
Input: Select-match subtree as smt, Query context node of smt as m, New query context node of smt as n, New binding bariable as bv', Binding variable map as $bvmap$.
Output: Unbound query for smt as q, New binding variable map as $bvmap'$.

1: $q \leftarrow UNBIND(m, n)$
2: $n_j \leftarrow$ the lowest common ancestor of m and n
3: $child_n(n_j) \leftarrow$ child node of n_j along the path from n_j to n
4: $R \leftarrow$ {nodes along the path from $child_n(n_j)$ to n}
5: **for** node $p \in R$ **do**
6: add the $SELECT$ columns of $Q_{bv(p)}$ to q
7: $P \leftarrow$ {nodes along the path from root of smt to m}
8: **for** node $p \in P$ **do**
9: **for** each child node c of p, such that $c \notin P$ and $c \notin R$ **do**
10: $q_c \leftarrow NEST(c, NULL)$
11: add $EXISTS$ q_c in $WHERE$ clause of q
12: $bvmap' \leftarrow bvmap$
13: **for** node $p \in R$ **do**
14: $bvmap'.insert(bv(p), bv')$
15: $child_m(n_j) \leftarrow$ child node of n_j along the path from n_j to m
16: $S \leftarrow$ {nodes along the path from $child_m(n_j)$ to m}
17: **for** node $s \in S$ **do**
18: $bvmap'.remove(bv(s))$
19: **return** $(q, bvmap')$

FIG. 17

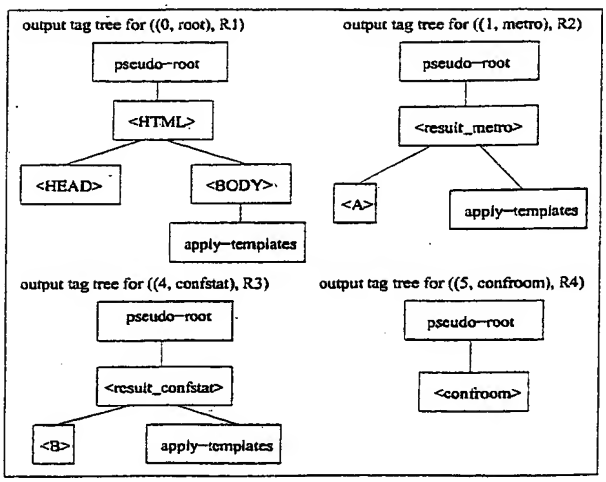


FIG. 18

(R1), (R3), and (R4) are the same as Figure 4.

```
<xsl:template match="metro">                    (R2)
  <xsl:apply-templates select="hotel/confstat"/>
</xsl:template>
```

FIG. 19

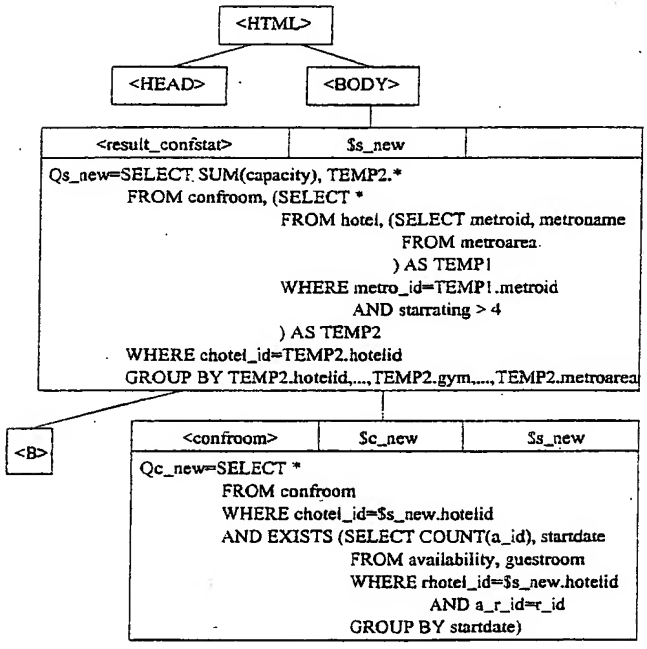


FIG. 20

```
(R1) and (R2) are the same as Figure 4.

<xsl:template match="confstat">                    (R3)
  <result_confstat>
    <B/>
    <xsl:apply-templates select=".[@sum<200]/
      ../hotel_available/../confroom
      [../confstat[@sum>100]][@capacity>250]"/>
  </result_confstat>
</xsl:template>

<xsl:template match="metro[@metroname=
          "chicago"]/hotel/confroom">    (R4)
    <xsl:value-of select="."/>
</xsl:template>
```

FIG. 21

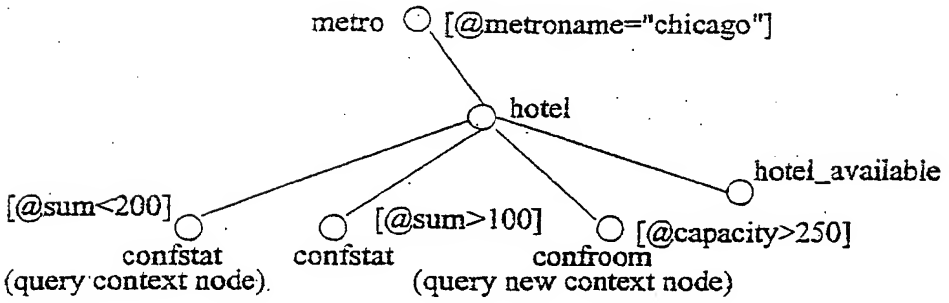


FIG. 22

{**Insert the following**
(a) after line 8 of Figure 13
(b) after line 1 of Figure 11}
$e_p \leftarrow$ predicate using $tag(p)$
add e_p in *WHERE* clause of q

FIG. 23

```
SELECT *  FROM confroom
WHERE chotel_id=$s_new.hotelid
      AND capacity > 250
      AND $s_new.SUM_capacity<200
      AND $s_new.metroname=''chicago''
      AND EXISTS (SELECT SUM(capacity)
                  FROM confroom,
                  WHERE chotel_id=$s_new.hotelid
                  HAVING SUM(capacity)>100)
      AND EXISTS (SELECT COUNT(a_id), startdate
                  FROM availability, guestroom
                  WHERE rhotel_id=$s_new.hotelid
                        AND a_r_id=r_id
                  GROUP BY startdate)
```

FIG. 24

```
<xsl:template match="pattern" mode="m">
  <xsl:if test="expression">
    template body
  </xsl:if>
</xsl:template>
```

(a)

```
<xsl:template match="pattern" mode="m">
  <xsl:apply-templates se-
lect=".[expression]" mode="mnew"/>
</xsl:template>

<xsl:template match="nodename" mode="mnew">
  template body
</xsl:template>
```

(b)

FIG. 25

```
<xsl:template match="pattern" mode="m">
  <xsl:choose>
    <xsl:when test="e1">b1</xsl:when>
    <xsl:when test="e2">b2</xsl:when>
    <xsl:otherwise>b3</xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

(a)

```
<xsl:template match="pattern" mode="m">
  <xsl:apply-templates
    select=".[e1]" mode="mnew1"/>
  <xsl:apply-templates
    select=".[not(e1) and e2]"
    mode="mnew2"/>
  <xsl:apply-templates
    select=".[not(e1) and not(e2)]"
    mode="mnew3"/>
</xsl:template>

<xsl:template match="nodename"
    mode="mnew1">b1</xsl:template>
<xsl:template match="nodename"
    mode="mnew2">b2</xsl:template>
<xsl:template match="nodename"
    mode="mnew3">b3</xsl:template>
```

(b)

FIG. 26

```
<xsl:template match="pattern" mode="m">
  <xsl:value-of select="expression"/>
</xsl:template>
```

(a)

```
<xsl:template match="pattern" mode="m">
  <xsl:apply-templates
     select="path expression" mode="mnew"/>
</xsl:template>

<xsl:template match="nodename[predicate]"
     mode="mnew">
  <xsl:value-of select="."/>
</xsl:template>
```

(b)

FIG. 27

```
<xsl:template match="pattern 1" mode="m">
  template body 1
</xsl:template>

<xsl:template match="pattern 2" mode="m">
  template body 2
</xsl:template>
```

(a)

```
<xsl:template match="pattern 1" mode="m1">
  template body 1
</xsl:template>

<xsl:template match="pattern 2" mode="m">
  <xsl:choose>
    <xsl:when test="expression 1">
      <xsl:apply-templates select="."
           mode="m1"/>
    </xsl:when>
    <xsl:otherwise>
      template body 2
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

(b)

FIG. 28

```
<xsl:template match="/metro">                    (R1)
  <xsl:param name="idx" select="10"/>
  <result_metro>
    <xsl:apply-templates
     select="hotel/hotel_available[@count>10]
             /metro_available[@count<$idx]">
      <xsl:with-param name="idx"
              select="$idx"/>
    </xsl:apply-templates>
  </result_metro>
</xsl:template>

<xsl:template match="metro_available">           (R2)
  <xsl:param name="idx"/>
  <xsl:choose>
    <xsl:when test="$idx<=1">
      <xsl:value-of select="."/>
    </xsl:when>
    <xsl:otherwise>
      <result_metroavail>
        <xsl:apply-templates
             select="self::[@count>50]/../../..">
          <xsl:with-param name="idx"
                          select="$idx-1"/>
        </xsl:apply-templates>
      <result_metroavail>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

FIG. 29

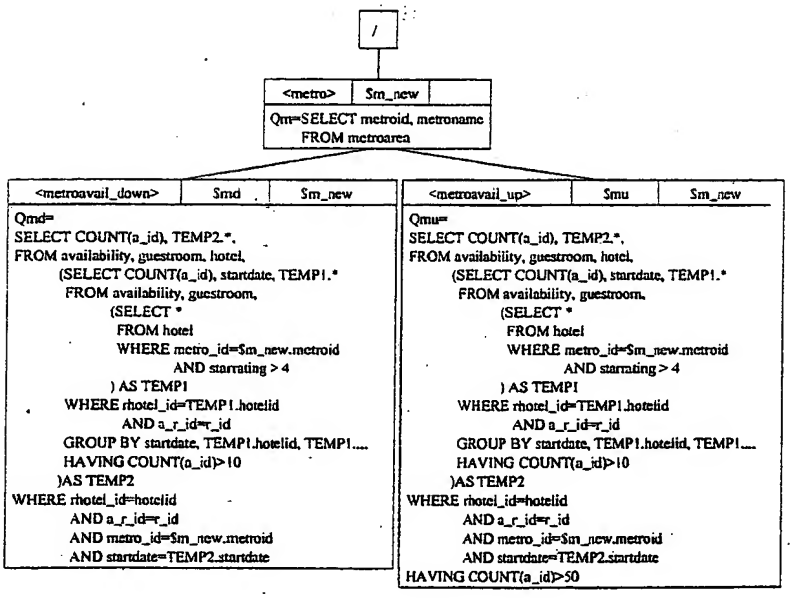


FIG. 30

```
<xsl:template match="/metro">                                    (R1')
    <xsl:param name="idx" select="10"/>
    <result_metro>
        <xsl:apply-templates
            select="metroavail_down[@count<$idx]">
          <xsl:with-param name="idx"
            select="$idx"/>
        </xsl:apply-templates>
    </result_metro>
</xsl:template>

<xsl:template match="metroavail_down">                           (R2')
    <xsl:param name="idx"/>
    <xsl:choose>
        <xsl:when test="$idx<=1">
            <xsl:value-of select="."/>
        </xsl:when>
        <xsl:otherwise>
            <result_metroavail>
                <xsl:apply-templates
                    select="../metroavail_up">
                  <xsl:with-param
                    name="idx" select="$idx-1"/>
                </xsl:apply-templates>
            <result_metroavail>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<xsl:template match="metroavail_up">                             (R3')
    <xsl:param name="idx"/>
        <result_metro>
            <xsl:apply-templates
                select="../metroavail_down[@count<$idx]">
                <xsl:with-param
                    name="idx" select="$idx"/>
            </xsl:apply-templates>
        <result_metro>
</xsl:template>
```

FIG. 31